SPATIALLAM: LATENT ACTIONS NEED SPATIAL STRUCTURE

Anonymous authors
Paper under double-blind review

ABSTRACT

Latent action models (LAMs) learn compact representations of visual transitions, enabling policies and world models to learn from videos with little or no action annotation. However, most existing LAMs represent latent actions as spatially unstructured tokens, leaving their correspondence to the locations of visual changes implicit. This provides limited structural support for reusing local transition content across positions, especially in tasks that require predicting or controlling similar local interactions at different image locations. To address this limitation, we propose SpatialLAM, which organizes latent actions on a 2D grid aligned with the observation plane and uses locality-biased attention to ground each token in a consistent visual neighborhood. We evaluate SpatialLAM in both policy learning and world modeling under matched data, optimization, and latent capacity. SpatialLAM improves policy success from 76.8% to 83.0% on LIBERO-Long, from 64.6% to 70.7% on out-of-distribution LIBERO-Plus, and from 45.0% to 66.7% across three real-world manipulation tasks. It also reduces world-model FVD by 16.9% on Something-Something V2 and 10.9% on Bridge V2. These results suggest that the utility of latent actions depends not only on compactly encoding visual transitions, but also on explicitly organizing their spatial structure.

1 INTRODUCTION

The high cost of collecting action-labeled robot demonstrations limits their scale and diversity (Brohan et al., 2023; Walke et al., 2023; Open X-Embodiment Collaboration et al., 2024; Khazatsky et al., 2024), motivating efforts to learn from more abundant human and robot videos (Goyal et al., 2017; Grauman et al., 2022). These videos, however, typically lack the action signals needed for direct control supervision (Baker et al., 2022). Latent action models (LAMs) address this gap by inferring compact action representations from consecutive observations (Schmidt & Jiang, 2024; Bruce et al., 2024), which can then serve as pseudo-action labels for policy learning or conditioning signals for world models (Ye et al., 2025; Bu et al., 2025; Gao et al., 2025). The inferred latents thus become an interface that downstream models must predict and interpret, making their organization important to how effectively these models can learn from visual transitions.

Most existing LAMs, however, organize this interface as spatially unstructured tokens. Prior work refines the information these tokens retain through bottleneck regularization (Schmidt & Jiang, 2024; Bruce et al., 2024; Ye et al., 2025; Gao et al., 2025), suppression of task-irrelevant dynamics (Nikulin et al., 2025; Bu et al., 2025; Yang et al., 2026; Bu et al., 2026), or transition factorization (Jeong et al., 2026; Klepach et al., 2026; Nam et al., 2026), but does not necessarily establish a stable correspondence between tokens and image regions. Without this correspondence, the same local interaction can produce unrelated latent patterns as its position changes, making it harder for policies and world models to reuse local transition content across locations (Zeng et al., 2020). We therefore seek position-aligned latent actions: translating a local interaction should shift its latent support while preserving the transition content it represents.

Our controlled analyses illustrate where this spatial reuse breaks down. When the same end-effector actions are translated across image locations, the unstructured baseline produces latent patterns with little systematic correspondence to the translation (in Figure 1). Consistent with this observation, policies trained with these latents also show weaker transfer to held-out positions, especially when training positions are sparse (in Figure 4). Query-level interventions further reveal that individual

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

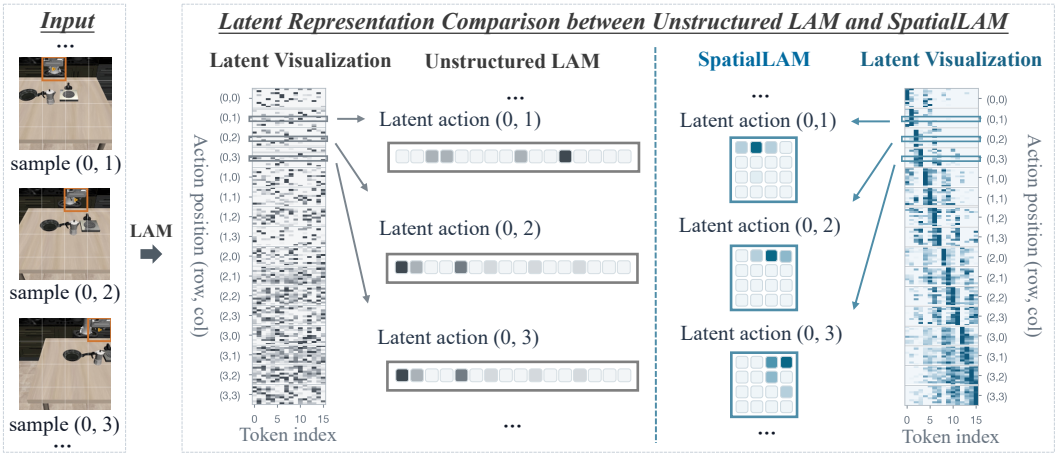


Figure 1: **Spatial organization improves cross-position consistency of latent actions.** We translate 10 end-effector actions to 16 image locations in a LIBERO kitchen tabletop environment (Liu et al., 2023), yielding 160 controlled samples. Each heatmap row visualizes the normalized latent action for one action at one location. In the unstructured baseline, changes in the latent pattern show little systematic correspondence with the underlying translation. In SpatialLAM, the latent pattern shifts across the grid with the interaction while retaining a consistent local structure.

tokens can affect localized regions, yet the region associated with a given token varies across samples (in Figure 5). The issue is therefore not simply whether a token captures a local effect, but whether its spatial meaning remains consistent across transitions.

To make this spatial meaning explicit, we propose SpatialLAM, which retains the standard inverse-forward LAM framework while arranging latent actions on a 2D grid aligned with the observation plane. Locality-biased attention grounds each token in a consistent visual neighborhood across samples, associating its grid position with where a transition occurs and its features with the local transition content. In our controlled translation experiment, this organization produces the intended behavior: moving an interaction shifts its latent pattern across the grid while preserving a consistent local structure (Figure 1). SpatialLAM thus provides an explicit spatial correspondence that downstream models can exploit when learning from interactions at different locations.

We evaluate whether this structured interface improves both policy learning and world modeling under matched data, optimization, and latent capacity. SpatialLAM raises policy success from 76.8% to 83.0% on LIBERO-Long, from 64.6% to 70.7% on out-of-distribution LIBERO-Plus, and from 45.0% to 66.7% across three real-world manipulation tasks. It also reduces world-model FVD by 16.9% on Something-Something V2 and 10.9% on Bridge V2. Controlled ablations further support the contribution of persistent spatial correspondence to the policy gains beyond locality alone. These results highlight spatial organization as a useful design dimension for latent actions, complementary to bottleneck capacity and transition reconstruction.

2 METHOD

SpatialLAM uses an inverse dynamics model to infer latent actions from two consecutive observations and a forward dynamics model to predict the next-frame features given the current-frame features and the inferred actions. Within this formulation, SpatialLAM introduces a fixed 2D latent grid, locality-biased attention, and a convolutional projector to establish consistent spatial grounding and preserve local adjacency. Figure 2 illustrates the overall framework.

Next, we first review the latent action learning formulation in Section 2.1 and then elaborate on the key components in Section 2.2.

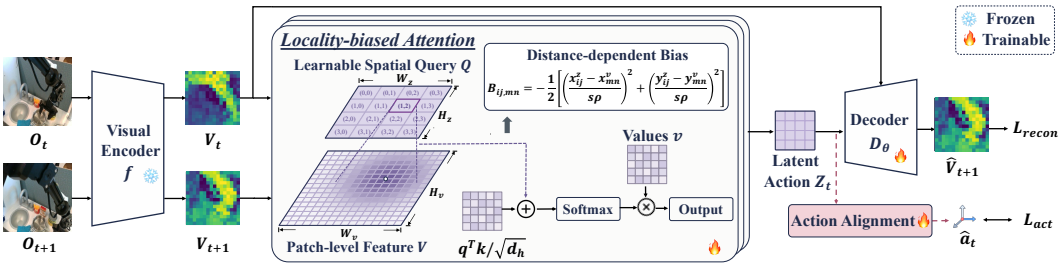


Figure 2: **Overview of SpatialLAM.** SpatialLAM infers a spatially structured latent action from consecutive visual features and learns it through future-state reconstruction and auxiliary action alignment. Locality-biased attention anchors each latent query to nearby visual evidence, promoting persistent token-to-region correspondence.

2.1 PRELIMINARIES

Given consecutive visual features $(\mathbf{V}_t, \mathbf{V}_{t+1})$ without an action label, a LAM learns a compact latent action $\mathbf{Z}_t$ that explains the transition between them. It couples an inverse dynamics model (IDM) and a forward dynamics model (FDM). The IDM encoder E_ϕ sees both $\mathbf{V}_t$ and $\mathbf{V}_{t+1}$ and infers $\mathbf{Z}_t$, while the FDM decoder D_θ sees only $\mathbf{V}_t$ and $\mathbf{Z}_t$ and predicts $\mathbf{V}_{t+1}$:

$$\mathbf{Z}_t = E_\phi(\mathbf{V}_t, \mathbf{V}_{t+1}), \qquad \hat{\mathbf{V}}_{t+1} = D_\theta(\mathbf{V}_t, \mathbf{Z}_t). \tag{1}$$

The two modules are trained jointly by reconstructing the future visual feature:

$$\mathcal{L}_{\text{recon}} = \left\| \hat{\mathbf{V}}_{t+1} - \mathbf{V}_{t+1} \right\|_2^2. \tag{2}$$

Observation encoding. Following recent work (Bu et al., 2025; Lee et al., 2026), we formulate latent action learning in the representation space of a frozen pretrained visual encoder, which reduces the need for the latent bottleneck to preserve low-level appearance details. Specifically, a frozen visual encoder f (*e.g.*, DINOv3 (Siméoni et al., 2025)) maps each frame to patch-level features:

$$\mathbf{V}_t = f(\mathbf{o}_t) \in \mathbb{R}^{N \times d_v}, \tag{3}$$

where d_v is the feature dimension of $\mathbf{V}_t$ and $N = H_v \times W_v$ patches lie on a two-dimensional spatial grid, with each patch corresponding to a fixed region of the image plane.

Compact bottleneck. To avoid a shortcut in which the learned latent directly carries future-frame appearance, LAMs restrict the transition through a compact bottleneck. Recent LAMs typically implement this bottleneck with a small set of learnable query tokens $\mathbf{Q} \in \mathbb{R}^{L_z \times d}$, where $L_z \ll N$ and d is the LAM width (Chen et al., 2025b; Bu et al., 2025; Gao et al., 2025). In the encoder E_ϕ shared by all our variants, the patch features of each frame are projected to width d, learnable query $\mathbf{Q}$ is prepended to each frame patches, and Transformer blocks update the query and patch tokens through within-frame self-attention, temporal attention across the two frames:

$$\big(\tilde{\mathbf{Q}}_t,\ \tilde{\mathbf{Q}}_{t+1}\big) \ = \ E_\phi\big([\,\mathbf{Q};\ \mathbf{V}_t\,],\ [\,\mathbf{Q};\ \mathbf{V}_{t+1}\,]\big). \tag{4}$$

The tilde denotes the updated query tokens for each frame. The updated patch tokens are discarded, and linear posterior heads map the queries $\tilde{\mathbf{Q}}_{t+1}$ of $\mathbf{V}_{t+1}$ to a d_z-dimensional bottleneck, producing $\mathbf{Z}_t \in \mathbb{R}^{L_z \times d_z}$ with $d_z \ll d$.

Training objective. Following AdaWorld (Gao et al., 2025), the IDM encoder is parameterized as a β-VAE (Kingma & Welling, 2014; Higgins et al., 2017), producing a diagonal Gaussian posterior $q_\phi(\mathbf{Z}_t \mid \mathbf{V}_t, \mathbf{V}_{t+1}) = \mathcal{N}\big(\boldsymbol{\mu}_t,\ \mathrm{diag}(\boldsymbol{\sigma}_t^2)\big)$. To ground the latent space in robot control and encourage the representation to capture action-relevant dynamics, a lightweight linear action alignment head ψ maps $\mathbf{Z}_t$ to the ground-truth action on a small labeled subset (Nikulin et al., 2025; Bi et al., 2025). The full objective combines feature-level reconstruction, KL regularization, and action alignment:

$$\mathcal{L} \ = \ \mathcal{L}_{\text{recon}} \ + \ \lambda_a\, \mathcal{L}_{\text{act}}\big(\mathbf{a}_t,\ \psi(\mathbf{Z}_t)\big) \ + \ \beta\, \mathrm{KL}\big[\, q_\phi(\mathbf{Z}_t \mid \mathbf{V}_t, \mathbf{V}_{t+1}) \,\|\, p(\mathbf{Z}_t)\,\big], \tag{5}$$

where $p(\mathbf{Z}_t) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a standard Gaussian prior, and λ_a and β control the strengths of action alignment and KL regularization, respectively. The action loss $\mathcal{L}_{\text{act}}$ applies SmoothL1 to the continuous translation and rotation components and binary cross-entropy to the gripper state. During LAM training, $\mathbf{Z}_t$ is a reparameterized sample from q_ϕ. All downstream models take the posterior mean $\bar{\mathbf{Z}}_t = \boldsymbol{\mu}_t$ as a deterministic latent action. While the compact bottleneck limits the information available to the latent representation, it places no constraint on how that information is spatially organized across latent tokens, allowing their correspondence to visual regions to vary across transitions.

2.2 Spatially Structured Latent Action Model

SpatialLAM addresses this limitation by changing the organization and downstream use of the same latent tokens. It keeps the inverse-forward training objective and modifies three interfaces: the latent query layout, the attention between latent and visual tokens, and the downstream latent projector.

Spatial action grid. First, SpatialLAM replaces the unstructured query set with a two-dimensional query grid of size $H_z \times W_z$ aligned with the observation patch grid $H_v \times W_v$. Each query token $\mathbf{Q}_{ij}$ receives a 2D sinusoidal positional encoding for its grid coordinate (i, j) (Vaswani et al., 2017; He et al., 2022). We set $L_z = H_z W_z$, so the inferred latent action is reshaped from $\mathbb{R}^{L_z \times d_z}$ into $\mathbf{Z}_t \in \mathbb{R}^{H_z \times W_z \times d_z}$, with $\mathbf{Z}_{t,ij}$ denoting the token at grid position (i, j). This grid establishes explicit spatial topology among latent tokens, but does not ensure correspondence with local visual regions.

Locality-biased attention. Second, SpatialLAM grounds each grid token by biasing attention toward nearby visual patches. Rather than masking distant patches, we use a soft locality bias to favor nearby visual evidence while preserving long-range attention.

Let $\mathbf{Q}_{ij}$ denote the latent token at row i and column j, anchored at $\mathbf{p}^z_{ij} = (x^z_{ij}, y^z_{ij})$, and let $\mathbf{V}_{mn}$ denote the visual patch token at $\mathbf{p}^v_{mn} = (x^v_{mn}, y^v_{mn})$, with all coordinates measured in patch units. Following standard scaled dot-product attention (Vaswani et al., 2017), we compute the query $\mathbf{q}_{ij} = \mathbf{W}_q \mathbf{Q}_{ij}$ and the key $\mathbf{k}_{mn} = \mathbf{W}_k \mathbf{V}_{mn}$, then add a distance-dependent bias to the pre-softmax logit:

$$\alpha_{ij,mn} = \frac{\mathbf{q}_{ij}^\top \mathbf{k}_{mn}}{\sqrt{d_h}} + B_{ij,mn}, \tag{6}$$

$$B_{ij,mn} = -\frac{1}{2}\left[\left(\frac{x^z_{ij} - x^v_{mn}}{s\rho}\right)^2 + \left(\frac{y^z_{ij} - y^v_{mn}}{s\rho}\right)^2\right], \tag{7}$$

where d_h is the attention-head dimension and, for our square grids, $s = H_v/H_z = W_v/W_z$ is the latent-grid spacing in patches. The locality scale ρ is measured in latent-grid spacings. A smaller ρ concentrates each query on nearby patches, while a larger ρ approaches global attention.

Because latent and patch tokens within each frame share the same self-attention, the softmax for $\mathbf{Q}_{ij}$ is taken over all tokens in that frame. The transposed bias $B_{ij,mn}$ is added to the patch-to-latent logits in the same attention. We apply the bias in every encoder block, independently on both input frames, and add its transposed form to the cross-attention of every decoder block. Patch-to-patch attention and temporal attention across frames remain unbiased.

Convolutional latent projector. Third, SpatialLAM keeps the grid structure at the downstream interface. Instead of flattening the latent action immediately, we apply a lightweight 3×3 convolution on the $H_z \times W_z$ latent grid before the downstream model consumes it. This preserves local adjacency and allows neighboring latent tokens to exchange information through a shared local operation.

Together, the spatial grid defines token topology, locality-biased attention grounds tokens to visual neighborhoods, and the convolutional projector exposes this adjacency to downstream models.

3 Experiments

3.1 Experimental Setup

Latent action models. We train all LAM variants on the same mixture of 27 Open X-Embodiment datasets (Open X-Embodiment Collaboration et al., 2024). Each frame is encoded by a frozen

Table 1: **Policy performance on LIBERO.** Success rate (%) across the four suites. Avg. denotes the arithmetic mean over the four suites. Unstructured LAM and SpatialLAM are evaluated under the same training and evaluation protocol, and their entries report the mean ± sample standard deviation over three seeds. Results for prior methods are taken from Lee et al. (2026). Best between unstructured LAM and SpatialLAM is shown in bold.

Method	Spatial	Object	Goal	Long	Avg.
Octo (Octo Model Team et al., 2024)	78.9	85.7	84.6	51.1	75.1
OpenVLA (Kim et al., 2024)	84.7	88.4	79.2	53.7	76.5
LAPA (Ye et al., 2025)	73.8	74.6	58.8	55.4	65.7
π_0 (Black et al., 2025)	96.8	98.8	95.8	85.2	94.2
UniVLA (Bu et al., 2025)	95.2	95.4	91.9	87.5	92.5
MVP-LAM (Lee et al., 2026)	96.0	94.6	94.8	90.8	94.1
Unstructured LAM	$96.6_{\pm 0.7}$	$99.8_{\pm 0.2}$	$\mathbf{97.4}_{\pm 0.7}$	$76.8_{\pm 2.2}$	$92.7_{\pm 0.5}$
SpatialLAM	$\mathbf{98.2}_{\pm 1.1}$	$\mathbf{100.0}_{\pm 0.0}$	$\mathbf{97.4}_{\pm 1.4}$	$\mathbf{83.0}_{\pm 1.8}$	$\mathbf{94.7}_{\pm 0.4}$

DINOv3-L/16 (Siméoni et al., 2025) into a 16×16 grid of patch features with $d_v = 1024$. The LAM encoder and decoder each consist of 8 Transformer blocks of width $d = 512$, and the latent bottleneck contains $L_z = 16$ tokens of dimension $d_z = 4$. All models are trained for 40k steps with a global batch size of 256. For SpatialLAM, the 16 latent tokens are arranged on a 4×4 spatial grid with locality scale $\rho = 1$. We refer to the spatially unstructured baseline as unstructured LAM. Unless otherwise specified, the unstructured LAM shares the same data, Transformer backbone, action alignment, optimizer, and latent capacity as SpatialLAM. The only architectural differences are the spatial action grid, locality-biased attention, and convolutional projector used by SpatialLAM. Detailed LAM and data settings for all variants are reported in Appendices A.1 and A.2.

Policy. We build the downstream policy upon the BAKU framework (Haldar et al., 2024), which encodes visual observations, proprioception, and language into a shared context and predicts temporally chunked robot actions. We augment BAKU with a latent head that predicts the frozen LAM representation from this context. The backbone and latent head are first pretrained on Bridge V2 (Walke et al., 2023) with the latent action objective alone. The full policy is then fine-tuned on each LIBERO suite (Liu et al., 2023), using 50 demonstrations per task. Appendix A.3 gives the formulation and training schedules. For Tables 1 and 2, we repeat the full unstructured-LAM and SpatialLAM pipelines with three seeds per method.

World model. Following prior latent action world models (Bruce et al., 2024; Gao et al., 2025; Zhang et al., 2026; Wang et al., 2026), we use the learned latent actions as control signals for future-state prediction. The world model predicts next-state features in the frozen DINOv3-L/16 representation space with 8 Transformer blocks of width 768, which apply self-attention over state patch tokens and cross-attention to latent action tokens. It is first trained on teacher-forced one-step prediction and then fine-tuned on 8-step autoregressive rollouts to match the inference condition. At inference, the frozen LAM encoder extracts latent actions from a reference video, and each autoregressive step from the initial observation is conditioned on the corresponding latent action. Appendix A.4 gives the formulation and world-model configurations.

3.2 Policy Evaluation

We evaluate whether the spatial organization of SpatialLAM improves robot control in three settings: standard LIBERO manipulation, LIBERO-Plus distribution shifts, and real-world manipulation. The unstructured LAM and SpatialLAM use the same policy architecture and training recipe, differing only in the latent action representation and its projector.

In-distribution control. LIBERO (Liu et al., 2023) comprises four language-conditioned manipulation suites: LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long. As shown in Table 1, SpatialLAM improves the average success rate over the unstructured LAM by $+2.0$ pp ($92.7\% \rightarrow 94.7\%$), with the largest gain on the LIBERO-Long suite ($+6.2$ pp, $76.8\% \rightarrow 83.0\%$).

Table 2: **Out-of-distribution policy performance on LIBERO-Plus.** Success rate (%) under seven perturbation factors. Prior-work results are taken from the official LIBERO-Plus benchmark (Fei et al., 2025). Unstructured LAM and SpatialLAM are evaluated by us under the same protocol, and their entries report the mean ± sample standard deviation over three seeds. Total is pooled over all 10,030 evaluation instances and is not the arithmetic mean of the seven columns. Best between unstructured LAM and SpatialLAM is shown in bold.

Method	Background Texture	Camera View	Lang. Instr.	Light Cond.	Object Layout	Robot Init.	Sensor Noise	Total
OpenVLA (Kim et al., 2024)	34.8	0.8	23.0	8.1	28.5	3.5	15.2	15.6
OpenVLA-OFT (Kim et al., 2025)	93.3	56.4	79.5	88.7	74.2	31.9	75.8	69.6
UniVLA (Bu et al., 2025)	81.0	1.8	69.6	69.0	31.9	46.2	21.2	42.9
π_0 (Black et al., 2025)	81.4	13.8	58.8	85.0	68.9	6.0	79.0	53.6
π_0-Fast (Pertsch et al., 2025)	73.2	65.1	61.0	73.2	68.8	21.6	74.4	61.6
Unstructured LAM	$39.5_{\pm 1.4}$	$80.9_{\pm 2.0}$	$70.0_{\pm 1.8}$	$80.2_{\pm 0.7}$	$69.4_{\pm 1.9}$	$16.1_{\pm 1.9}$	$91.1_{\pm 1.1}$	$64.6_{\pm 0.8}$
SpatialLAM	$\mathbf{46.9}_{\pm 2.6}$	$\mathbf{91.1}_{\pm 1.2}$	$\mathbf{70.4}_{\pm 0.8}$	$\mathbf{88.1}_{\pm 1.3}$	$\mathbf{74.2}_{\pm 2.6}$	$\mathbf{25.2}_{\pm 1.4}$	$\mathbf{94.9}_{\pm 1.3}$	$\mathbf{70.7}_{\pm 1.0}$

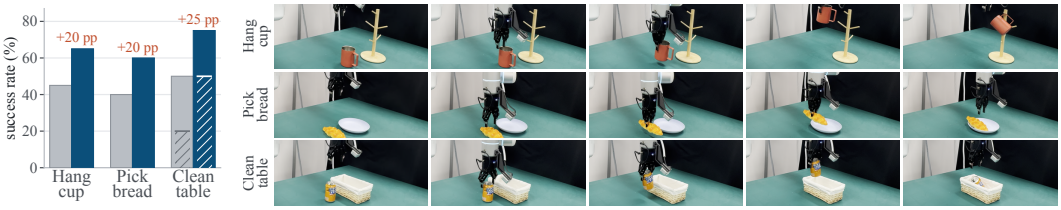


Figure 3: **Real-world manipulation.** Left, success rate of the unstructured LAM (gray) and SpatialLAM (blue) over 20 trials per task. Hatched bars in Clean table show the displaced-basket evaluation with one trial in each of ten directions. Right, successful SpatialLAM executions.

Out-of-distribution robustness. LIBERO-Plus (Fei et al., 2025) extends LIBERO with seven controlled distribution shifts that perturb the visual scene, the robot state, and the language instruction. As shown in Table 2, SpatialLAM improves upon the unstructured LAM on every perturbation factor and raises the total success rate from 64.6% to 70.7% (+6.1 pp). The largest gains appear under camera-view (+10.2 pp) and robot-initialization (+9.1 pp) shifts, both of which alter the spatial layout of the observation. SpatialLAM also yields consistent improvements under lighting, background, object-layout, and sensor-noise perturbations (+7.9, +7.4, +4.8, and +3.8 pp, respectively). Language perturbations are essentially unaffected (+0.4 pp), as expected.

Real-world manipulation. We evaluate three tasks on a real-world Flexiv setup: hanging a cup on a rack (Hang cup), placing bread on a plate (Pick bread), and placing a can into a basket (Clean table). For each task, both policies are fine-tuned on the same 30 teleoperated demonstrations and evaluated over 20 trials. To assess real-world cross-position generalization, we further evaluate Clean table under an out-of-distribution basket displacement, where the basket is translated by 5 cm from its training placement in ten directions. This creates relative basket-to-can configurations not observed during training. As shown in Figure 3, SpatialLAM improves success on all three tasks, increasing the pooled success rate from 45.0% to 66.7% (27/60 to 40/60 trials), and from 20% to 50% (2/10 to 5/10 trials) under basket displacement. Appendix A.6 provides full experiment details.

3.3 World-Model Evaluation

We next evaluate whether spatially structured latent actions provide better conditioning signals for predictive world models. We train otherwise identical world models conditioned on latent actions from the unstructured LAM or SpatialLAM. No-action and recorded robot-action conditioning serve as references, with recorded actions available only on Bridge V2. We evaluate autoregressive prediction on Bridge V2 robot videos (Walke et al., 2023) and Something-Something V2 (SSv2) human videos (Goyal et al., 2017). We report PSNR, LPIPS, and feature NMSE at one-step teacher forcing and at rollout steps 4 and 8, together with FVD (Unterthiner et al., 2018) over the full $H = 8$ rollout.

Table 3: **World-model prediction quality under latent-action conditioning.** *No Action* zeros the action input of an otherwise identical dynamics model. *Recorded Robot Action* replaces the latent action with the recorded robot commands, and is reported on Bridge only since SSv2 has no recorded actions. NMSE is normalized by the error of a persistence predictor that keeps the state feature unchanged. Best in bold.

Method	PSNR ↑			LPIPS ↓			NMSE ↓			FVD ↓
	H1	H4	H8	H1	H4	H8	H1	H4	H8	
Bridge										
No Action	16.86	15.91	15.27	0.241	0.276	0.306	0.737	0.808	0.847	1734.9
Recorded Robot Action	18.38	16.72	16.23	0.215	0.248	0.276	0.633	0.699	0.757	1130.4
Unstructured LAM	20.72	20.14	19.20	0.169	0.187	0.213	0.471	0.488	0.543	861.7
SpatialLAM	**21.06**	**20.44**	**19.47**	**0.163**	**0.179**	**0.203**	**0.450**	**0.465**	**0.508**	**767.7**
SSv2										
No Action	20.14	16.62	15.21	0.269	0.403	0.461	0.804	0.761	0.699	1408.8
Unstructured LAM	22.00	19.04	17.00	0.219	0.315	0.388	0.655	0.595	0.577	663.8
SpatialLAM	**22.12**	**19.39**	**17.36**	**0.206**	**0.301**	**0.371**	**0.648**	**0.582**	**0.562**	**551.3**

As shown in Table 3, action conditioning is essential for accurate prediction. On Bridge V2, FVD drops from 1734.9 without action conditioning to 1130.4 with recorded robot commands and further to 861.7 with the unstructured latent action, reflecting the richer transition information captured by latents inferred from realized visual changes. On SSv2, latent conditioning similarly reduces FVD from 1408.8 to 663.8. Among latent action models, SpatialLAM consistently improves PSNR, LPIPS, and NMSE at every evaluated horizon on both datasets. The NMSE gap grows with rollout horizon, from 0.021 at $H = 1$ to 0.035 at $H = 8$ on Bridge V2, and from 0.007 to 0.015 on SSv2. SpatialLAM also reduces FVD by 10.9% on Bridge V2 ($861.7 \rightarrow 767.7$) and 16.9% on SSv2 ($663.8 \rightarrow 551.3$). The consistent gains on SSv2 further suggest that the benefit of spatial structure extends beyond robot-specific dynamics to general manipulation videos.

3.4 Ablation Studies

We examine component contributions, cross-position transfer, and token-to-region correspondence. Unless otherwise stated, policy ablations use LIBERO-Long and a fixed, stratified 10% subset of LIBERO-Plus shared across variants. We refer to this subset as the Plus subset in the ablation table. World-model FVD uses eight-step rollouts. Additional ablations are provided in Appendix C.

Component and control ablation. Table 4 isolates the contributions of the SpatialLAM components under matched training recipes and latent capacity. Neither the spatial grid alone nor the convolutional projector applied to the unstructured baseline yields a consistent policy gain. Adding locality bias accounts for most of the improvement in the component ladder, raising success to 81.8% on LIBERO-Long and 68.2% on Plus subset. The convolutional projector adds another 1.2 and 1.4 pp, respectively. All latent variants improve policy success over the no latent baseline.

To disentangle persistent spatial correspondence from locality alone, the shuffled-grid variant retains the same locality bias but varies the query-to-anchor assignment across transitions. For each sample, it selects one of 16 fixed permutations according to the latent-grid position with the largest feature change. Queries therefore remain locally grounded within each transition, but their spatial correspondence vary across transitions. As in the unstructured LAM, the resulting latent remains a deterministic mapping of the input pair. Relative to + locality bias with the same per-token linear projector, shuffling reduces success from 81.8% to 76.0% on LIBERO-Long and from 68.2% to 64.6% on Plus subset. This suggests that locality alone is insufficient to explain the policy gains and that transfer also benefits from stable token-to-region correspondence across transitions.

Under the same latent budget and training recipe, OTF-LAM (Nam et al., 2026), which factorizes local transition effects, and PoLAR (Jeong et al., 2026), which factorizes transition extent and mode, both improve policy success and SSv2 FVD over the unstructured baseline but remain below SpatialLAM on all reported metrics. SpatialLAM adds negligible inference cost, increasing both

Table 4: **Component and control ablation.** All variants share training data, optimization, and latent capacity. Success is reported on LIBERO-Long and Plus subset, and FVD uses $H = 8$ rollouts.

Method	Success (↑)		FVD (↓)	
	Long	Plus	Bridge	SSv2
Component ladder				
Unstructured LAM	76.8	63.2	861.7	663.8
+ spatial grid	77.2	62.8	862.8	652.7
+ locality bias	81.8	68.2	797.1	572.4
+ conv. proj. (SpatialLAM)	**83.0**	**69.6**	**767.7**	**551.3**
Controls				
No latent	72.4	58.5	–	–
Unstructured + conv. proj.	77.0	62.6	829.2	668.4
Shuffled grid + locality bias	76.0	64.6	846.4	601.2
External baselines				
OTF-LAM (Nam et al., 2026)	78.4	66.2	854.8	621.6
PoLAR (Jeong et al., 2026)	77.6	65.9	864.4	636.1

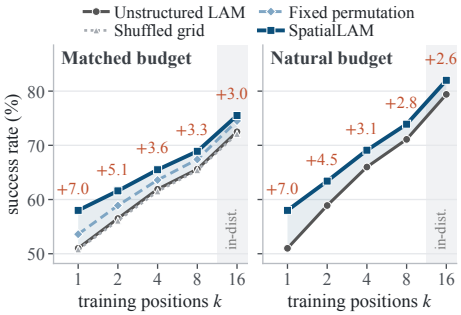


Figure 4: **Cross-position spatial generalization on translated LIBERO-Long.** Policies are trained on $k \in \{1, 2, 4, 8, 16\}$ positions and evaluated on eight fixed positions. Evaluation positions are held out for $k \leq 8$ and included for the $k = 16$ reference. Matched budget fixes the total demonstrations, while natural budget uses all demonstrations from selected positions. Each point uses 800 rollouts.

parameter count and FLOPs by less than 0.02% when measured over the LAM plus downstream latent projector. Further experimental and compute details are provided in Appendix B.2.

Cross-position spatial generalization. As shown in Figure 4, we train policies on k of 16 translated LIBERO-Long spatial positions and evaluate them on eight fixed evaluation positions, with each LAM frozen. Under a matched budget, which fixes the total number of demonstrations, the SpatialLAM advantage is largest at $k = 1$ ($+7.0$ pp) and narrows as spatial coverage grows, remaining $+3.0$ pp at $k = 16$, where the evaluation positions are all seen in training. A natural budget, in which the number of demonstrations grows with k, shows the same trend ($+7.0$ to $+2.6$ pp). SpatialLAM therefore generalizes better to held-out positions under sparse spatial supervision, and its benefit persists when all positions are covered.

The two correspondence controls separate persistence from geometry. Both keep the locality bias and the 3×3 projector of SpatialLAM and change only the query-to-anchor assignment. The shuffled grid, whose assignment varies with the transition, stays at the unstructured level for every k. A fixed permutation, which keeps each query at a persistent but spatially scrambled anchor, improves over the shuffled grid at every k but remains below SpatialLAM, whose anchors also preserve image-plane adjacency. The gap between the fixed permutation and SpatialLAM is largest at $k = 1$ and narrows as k grows. Persistence thus helps at every k, whereas geometric alignment contributes most when spatial coverage is sparse. Appendix C.3 further confirms that translating an interaction shifts the SpatialLAM latent across the grid. Detailed settings are provided in Appendix B.3.

Persistent spatial correspondence of latent tokens. Figure 5 visualizes the spatial influence of individual latent tokens through interventions on the FDM reconstruction. Unstructured latents already exhibit localized effects, but their associated regions vary across transitions. In contrast, SpatialLAM maintains stable token-to-region correspondence across transitions, with each token remaining associated with a consistent neighborhood around its prescribed anchor. These results show that SpatialLAM does more than induce locality: it organizes local transition effects into a persistent spatial correspondence that is preserved across transitions. Details are provided in Appendix B.4.

4 Related Work

Latent action representations. Latent Action Models (LAMs) bridge passive visual observation and active control by extracting action-like representations from videos (Schmidt & Jiang, 2024; Bruce et al., 2024). Subsequent work improves these representations by changing their information content or supervision. LAPA, Moto, and AdaWorld constrain the latent through discrete quantiza-

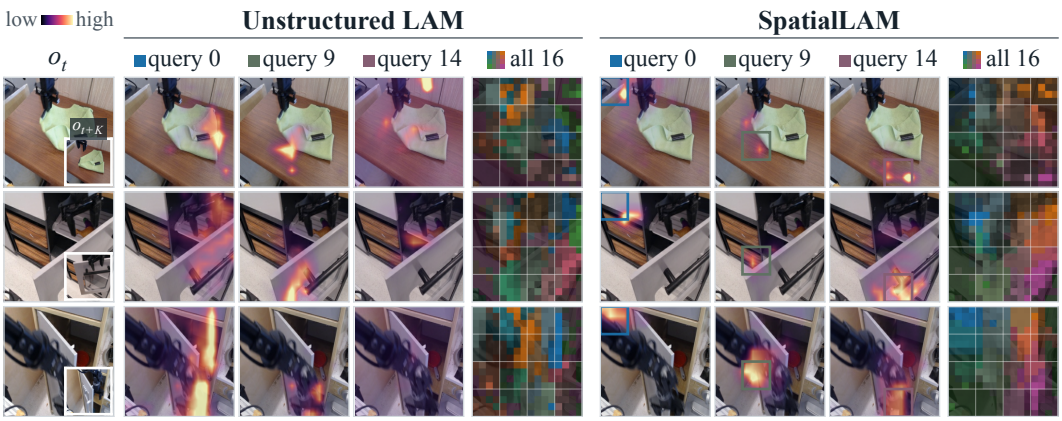


Figure 5: **Spatial influence of latent tokens.** Each map shows the residual change in FDM reconstruction after perturbing one latent token with the same-index token from another held-out transition. Boxes indicate the spatial region associated with the corresponding SpatialLAM anchor. The last column assigns each visual patch to its most influential token, revealing the spatial organization of token influence across the full latent set.

tion or compact continuous bottlenecks (Ye et al., 2025; Chen et al., 2025b; Gao et al., 2025), while LAOM, Motus, UniVLA, and villa-X add action supervision or task semantics to emphasize control-relevant dynamics (Nikulin et al., 2025; Bi et al., 2025; Bu et al., 2025; Chen et al., 2025a). Other methods introduce structure through motion- or flow-aligned objectives (Yang et al., 2026; Bu et al., 2026), cross-view reconstruction (Lee et al., 2026), and decompositions into local transition effects, transition extent and mode, or object-centric factors (Nam et al., 2026; Jeong et al., 2026; Klepach et al., 2026). SpatialLAM is complementary to these directions: it studies the spatial organization of the latent action interface itself and assigns latent tokens persistent image-plane roles.

Spatial representations for manipulation. Spatial correspondence between visual observations and manipulation has a long-standing history in robot learning (Finn et al., 2016; Florence et al., 2018). Transporter Networks and CLIPort preserve image-plane structure for spatially grounded control (Zeng et al., 2020; Shridhar et al., 2021), and locality-aware policies similarly use proximity between visual evidence and actions (Zhang et al., 2024). Other approaches represent actions through image-space quantities such as point tracks or dense optical flow (Wen et al., 2024; Bharadhwaj et al., 2024; Xu et al., 2024). SpatialLAM shares the use of image-plane structure, but it does not prescribe explicit trajectories, flow fields, or action points. Instead, it uses spatial indexing and local adjacency to organize a compact learned latent action.

5 Conclusion

We presented SpatialLAM, a spatially structured latent action interface for policy learning and world modeling. Its image-aligned 2D grid, locality-biased attention, and convolutional projector provide consistent spatial grounding while preserving local adjacency. Controlled diagnostics show that this organization allows latent patterns to shift systematically with local interactions while retaining their transition structure, supporting reuse across positions. Under matched training recipes and latent capacity, SpatialLAM improves policy performance on LIBERO, LIBERO-Plus, and real-world manipulation tasks, as well as action-conditioned world modeling on human and robot videos. These findings support spatial organization as an important design principle for latent actions alongside compactness and action relevance.

The current design anchors latent actions to the image plane and is evaluated primarily in static-camera settings, leaving its robustness to camera motion and viewpoint changes an open question. Future work should explore camera-aware, object-centric, and embodiment-aware coordinate systems, and examine whether the benefits of spatially structured latent actions extend to broader video sources and more diverse embodiments.

AI USE STATEMENT

In this work, we used generative AI tools as part of research execution to assist with code writing and debugging. We did not use generative AI tools for research ideation, experimental design, data generation, or interpretation of experimental results, and the remaining tasks requiring disclosure are not applicable to this work. Additionally, we used generative AI tools to assist with literature search and summarization and to polish the manuscript for clarity and readability. All AI-assisted work was reviewed by the authors. AI-assisted code was manually inspected and tested, and AI-polished manuscript content and literature summaries were checked for technical accuracy and consistency with the original sources. We take responsibility for the final content of this work, including text, claims, code, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed model architectures, training configurations, data processing procedures, downstream policy and world-model settings, and evaluation protocols in the main text and Appendix. The appendix further documents the implementation details and ablation settings used throughout our experiments.

REFERENCES

Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video PreTraining (VPT): Learning to act by watching unlabeled online videos. In *NeurIPS*, 2022.

Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2Act: Predicting point tracks from internet videos enables generalizable robot manipulation. In *ECCV*, 2024.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, et al. Motus: A unified latent action world model. *arXiv:2512.13030*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, et al. π_0: A vision-language-action flow model for general robot control. In *RSS*, 2025.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, et al. RT-1: Robotics transformer for real-world control at scale. In *RSS*, 2023.

Jake Bruce, Michael Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal Behbahani, Stephanie Chan, Nicolas Heess, Lucy Gonzalez, et al. Genie: Generative interactive environments. In *ICML*, 2024.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. UniVLA: Learning to act anywhere with task-centric latent actions. In *RSS*, 2025.

Xizhou Bu, Jiexi Lyu, Fulei Sun, Ruichen Yang, Zhiqiang Ma, and Wei Li. LAOF: Robust latent action learning with optical flow constraints. In *CVPR*, 2026.

Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

Xiaoyu Chen, Hangxing Wei, Pushi Zhang, Chuheng Zhang, Kaixin Wang, Yanjiang Guo, Rushuai Yang, Yucen Wang, Xinquan Xiao, Li Zhao, Jianyu Chen, and Jiang Bian. villa-X: Enhancing latent action modeling in vision-language-action models. *arXiv:2507.23682*, 2025a.

Yi Chen, Yuying Ge, Weiliang Tang, Yizhuo Li, Yixiao Ge, Mingyu Ding, Ying Shan, and Xihui Liu. Moto: Latent motion token as the bridging language for learning robot manipulation from videos. In *ICCV*, 2025b.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: In-depth robustness analysis of vision-language-action models. *arXiv:2510.13626*, 2025.

Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In *ICRA*, 2016.

Peter R Florence, Lucas Manuelli, and Russ Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. In *CoRL*, 2018.

Shenyuan Gao, Siyuan Zhou, Yilun Du, Jun Zhang, and Chuang Gan. AdaWorld: Learning adaptable world models with latent actions. In *ICML*, 2025.

Songwei Ge, Thomas Hayes, Harry Yang, Xi Yin, Guan Pang, David Jacobs, Jia-Bin Huang, and Devi Parikh. Long video generation with time-agnostic VQGAN and time-sensitive transformer. In *ECCV*, 2022.

Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzyńska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "Something Something" video database for learning and evaluating visual common sense. In *ICCV*, 2017.

Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, Miguel Martin, Tushar Nagarajan, Ilija Radosavovic, Santhosh Kumar Ramakrishnan, Fiona Ryan, Jayant Sharma, et al. Ego4D: Around the world in 3,000 hours of egocentric video. In *CVPR*, 2022.

Siddhant Haldar, Zhuoran Peng, and Lerrel Pinto. BAKU: An efficient transformer for multi-task policy learning. In *NeurIPS*, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β-VAE: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

Youngjoon Jeong, Jihwan Yu, Minsoo Jo, Junha Chun, and Taesup Kim. PoLAR: Factorizing extent and mode in latent actions for robot policy learning. In *CoRL*, 2026.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *RSS*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *CoRL*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *RSS*, 2025.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *ICLR*, 2014.

Albina Klepach, Alexander Nikulin, Ilya Zisman, Denis Tarasov, Alexander Derevyagin, Andrei Polubarov, Nikita Lyubaykin, Igor Kiselev, and Vladislav Kurenkov. Object-centric latent action learning. In *AAAI*, 2026.

Jung Min Lee, Dohyeok Lee, Seokhun Ju, Taehyun Cho, Jin Woo Koo, Li Zhao, Sangwoo Hong, and Jungwoo Lee. MVP-LAM: Learning action-centric latent action via cross-viewpoint reconstruction. *arXiv:2602.03668*, 2026.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 17(39), 2016.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *NeurIPS*, 2023.

Heejeong Nam, Chandradithya S. Jonnalagadda, Harshit Aggarwal, Eric Xu, and Randall Balestriero. Latent actions from factorized transition effects under agent ambiguity. *arXiv:2606.30544*, 2026.

Alexander Nikulin, Ilya Zisman, Denis Tarasov, Nikita Lyubaykin, Andrei Polubarov, Igor Kiselev, and Vladislav Kurenkov. Latent action learning requires supervision in the presence of distractors. In *ICML*, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *RSS*, 2024.

Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, et al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *ICRA*, 2024.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. In *RSS*, 2025.

Dominik Schmidt and Minqi Jiang. Learning to act without actions. In *ICLR*, 2024.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. CLIPort: What and where pathways for robotic manipulation. In *CoRL*, 2021.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, et al. DINOv3. *arXiv:2508.10104*, 2025.

Jaskirat Singh, Boyang Zheng, Zongze Wu, Richard Zhang, Eli Shechtman, and Saining Xie. Improved baselines with representation autoencoders. *arXiv:2605.18324*, 2026.

Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric and challenges. *arXiv:1812.01717*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. BridgeData V2: A dataset for robot learning at scale. In *CoRL*, 2023.

Junke Wang, Qihang Zhang, Shuai Yang, Yiming Luo, Yujun Shen, Zuxuan Wu, Yu-Gang Jiang, and Yinghao Xu. RepWAM: World action modeling with representation visual-action tokenizers. *arXiv:2606.13674*, 2026.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *NeurIPS*, 2020.

Chuan Wen, Xingyu Lin, John So, Kai Chen, Qi Dou, Yang Gao, and Pieter Abbeel. Any-point trajectory modeling for policy learning. In *RSS*, 2024.

Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018.

Mengda Xu, Zhenjia Xu, Yinghao Xu, Cheng Chi, Gordon Wetzstein, Manuela Veloso, and Shuran Song. Flow as the cross-domain manipulation interface. In *CoRL*, 2024.

Jiange Yang, Yansong Shi, Haoyi Zhu, Mingyu Liu, Kaijing Ma, Yating Wang, Gangshan Wu, Tong He, and Limin Wang. CoMo: Learning continuous latent motion from internet videos for scalable robot learning. In *CVPR*, 2026.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Sejune Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *ICLR*, 2025.

Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Ayzaan Wahid, Vikas Sindhwani, and Johnny Lee. Transporter networks: Rearranging the visual world for robotic manipulation. In *CoRL*, 2020.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Tianqiu Zhang, Muyang Lyu, Yufan Zhang, Fang Fang, and Si Wu. DiLA: Disentangled latent action world models. In *ICML*, 2026.

Tong Zhang, Yingdong Hu, Jiacheng You, and Yang Gao. Leveraging locality to boost sample efficiency in robotic manipulation. In *CoRL*, 2024.

A Implementation Details

A.1 Latent Action Model

Observation features. SpatialLAM operates on visual representations extracted by a frozen DINOv3-L/16 encoder (Siméoni et al., 2025). Input RGB frames are resized to 256×256, producing a 16×16 grid of 256 patch tokens with dimension 1024. Patch representations are obtained by summing intermediate tokens from every other layer in the upper half of the encoder, with CLS and register tokens discarded. The resulting features are standardized per channel using fixed statistics, and reconstruction is optimized with mean squared error in the standardized feature space. The visual encoder remains frozen throughout training.

Architecture. The LAM uses a hidden width of 512, 8 attention heads, an MLP ratio of 4, and no dropout. Features from both input frames are linearly projected to width 512 and augmented with fixed 2D sinusoidal patch positional encodings. Sixteen learned latent queries are prepended to each frame and augmented with fixed latent positional encodings, together with a learned two-frame temporal embedding. Each of the 8 encoder blocks applies spatial self-attention within each frame, temporal attention across the two frames at corresponding token positions, and an MLP. The posterior parameters are predicted from the latent queries of the second frame using linear heads for μ and $\log \sigma^2$, with $\log \sigma^2$ clamped to $[-10, 10]$.

The decoder takes the current-frame patch features as 256 spatial tokens and projects the 16 latent tokens to width 512 with the corresponding latent positional encodings. Each of its 8 blocks applies patch self-attention followed by patch-to-latent cross-attention and an MLP. The decoder predicts a gated residual update to the current feature map, yielding the reconstructed next-frame representation $\hat{\mathbf{V}}_{t+1}$.

Latent topology and locality bias. The unstructured LAM assigns the 16 latent queries fixed 1D sinusoidal positional encodings and applies no locality bias. SpatialLAM instead arranges the 16 queries on a 4×4 grid and assigns fixed 2D sinusoidal positional encodings. The grid is aligned with the 16×16 visual patch grid, with latent position (i, j) anchored at

$$\mathbf{p}^z_{ij} = (4j + 1.5,\ 4i + 1.5), \qquad i, j \in \{0, 1, 2, 3\}, \tag{8}$$

while visual patch (m, n) is located at $\mathbf{p}^v_{mn} = (n, m)$ for $m, n \in \{0, \ldots, 15\}$. These coordinates are in patch units. Equation 7 divides their differences by $s\rho$ with latent-grid spacing $s = H_v/H_z = 4$, so the default $\rho = 1$ corresponds to four patches.

Equation 6 shows the latent-to-patch direction. In the implementation, latent and patch tokens of the same frame share one spatial self-attention in every encoder block, whose softmax runs over all tokens of that frame. The bias $B_{ij,mn}$ is added to the latent-to-patch logits and, in transposed form, to the patch-to-latent logits. Latent-to-latent and patch-to-patch logits and the temporal attention across frames receive no bias. In every decoder block, the transposed bias is added to the patch-to-latent cross-attention. The same bias is shared across attention heads and is applied independently to both input frames.

Objective. The KL term in Equation 5 is summed over the 64 latent dimensions (16 tokens with width 4) and averaged over the batch. Its weight is fixed at $\beta = 2{\times}10^{-4}$ throughout training.

Table 5 lists the remaining architecture and optimization settings.

A.2 Training Data and Action Alignment

Mixture and frame-pair sampling. LAM training uses 27 Open X-Embodiment datasets (Open X-Embodiment Collaboration et al., 2024). Table 6 reports the mixture weight and fixed temporal frame gap used for each dataset. Training pairs are sampled within episodes using these dataset-specific gaps, which are kept identical across all LAM variants.

Action alignment. We additionally apply action supervision to 10% of the LAM training samples. Action-aligned samples are drawn from the 20 datasets in Table 6 that provide complete 7-DoF translation, rotation, and gripper annotations. For these samples, we use adjacent frame pairs $(\mathbf{o}_t, \mathbf{o}_{t+1})$

Table 5: Latent action model hyperparameters.

Architecture		*Optimization*	
Visual encoder	DINOv3-L/16, frozen	Steps	40,000
Input resolution	256×256	Global batch	256 pairs
Patch grid	16×16, dim 1024	Per-device batch	16
Width / heads	512 / 8	Gradient accumulation	4
Encoder / decoder blocks	8 / 8	Optimizer	AdamW $(0.9, 0.999)$
MLP ratio / dropout	4 / 0.0	Weight decay	10^{-2}
Latent tokens L_z	16	Learning rate	$10^{-4} \to 10^{-5}$, cosine
Latent width d_z	4	Warmup	2,000 steps
Locality scale ρ	1 grid spacing (4 patches)	Gradient clip	0.3
KL weight β	$2{\times}10^{-4}$	Precision	bf16
Action weight λ_a	10^{-2}	Hardware	4×H100-80GB
		Seeds	$\{42, 43, 44\}$

Table 6: **Open X-Embodiment mixture.** Weight is the configured mixture weight, p_{cfg} its normalized share, and gap the temporal distance in raw frames between the two frames of a pair. Datasets marked † provide complete end-effector action annotations and are used for action alignment.

Dataset	Weight	p_{cfg}	Gap	Dataset	Weight	p_{cfg}	Gap
fractal20220817†	0.5	1.56%	4	furniture_bench†	0.1	0.31%	9
kuka†	0.1	0.31%	9	ucsd_kitchen†	2.0	6.25%	2
bridge†	1.0	3.12%	9	austin_sailor†	1.0	3.12%	15
taco_play†	2.0	6.25%	9	austin_sirius†	1.0	3.12%	9
jaco_play	1.0	3.12%	9	dlr_edan†	1.0	3.12%	2
berkeley_cable	1.0	3.12%	9	iamlab_cmu†	1.0	3.12%	19
roboturk†	2.0	6.25%	9	utaustin_mutex†	1.0	3.12%	14
viola†	2.0	6.25%	14	berkeley_fanuc	2.0	6.25%	9
berkeley_autolab†	2.0	6.25%	2	cmu_stretch†	1.0	3.12%	9
toto†	1.0	3.12%	19	bc_z†	0.2	0.62%	9
language_table	0.1	0.31%	9	berkeley_gnm_recon	1.0	3.12%	9
stanford_hydra†	2.0	6.25%	9	berkeley_gnm_cory	1.0	3.12%	9
austin_buds†	1.0	3.12%	9	berkeley_gnm_sac_son	1.0	3.12%	9
nyu_franka_play†	3.0	9.38%	3				

and align the inferred latent action with the corresponding one-step robot action $\mathbf{a}_t$. Actions are expressed as end-effector displacements in the control frame of the corresponding embodiment. The translation and rotation components are normalized to $[-1, 1]$ using per-dataset first- and ninety-ninth-percentile statistics, while the gripper component is mapped to $[0, 1]$, with 1 denoting open.

The action alignment head ψ in Equation 5 is a linear layer that maps the flattened posterior latent, 16 tokens of width 4, to the 7-dimensional action. For SpatialLAM, this corresponds to the map $\mathbb{R}^{H_z \times W_z \times d_z} \to \mathbb{R}^7$ over the 4×4 latent grid. The alignment loss applies SmoothL1 to the six continuous dimensions and binary cross-entropy to the gripper, weighted by $\lambda_a = 10^{-2}$. Unstructured LAM and SpatialLAM use the same alignment data, alignment head, and weight. The action alignment head ψ is used only during LAM training and discarded afterwards.

A.3 Policy

In both pretraining and fine-tuning stages, the frozen LAM encoder provides the regression target $\bar{\mathbf{Z}}_t = \boldsymbol{\mu}_\phi\big(f(\mathbf{o}_t),\, f(\mathbf{o}_{t+T_a})\big)$, the posterior mean over a stride equal to the chunk length T_a. The target thus summarizes the transition from $\mathbf{o}_t$ to the observation reached after executing the action chunk $\mathbf{A}_t$. Both frames come from a single fixed third-person camera, the primary camera on Bridge V2 and the agent view on LIBERO. The LIBERO policy additionally observes the wrist view, which is not used for the target. The latent head h_z predicts the latent action $\hat{\mathbf{Z}}_t$ from the policy context. In pretraining, we optimize the backbone and h_z on Bridge V2 videos using only latent action

prediction, without action labels:

$$\mathcal{L}_{\text{pretrain}} = \text{SmoothL1}\left(\hat{\mathbf{Z}}_t,\, \text{sg}[\bar{\mathbf{Z}}_t]\right), \tag{9}$$

where $\text{sg}[\cdot]$ stops gradients through the LAM. In fine-tuning, we attach the action decoder h_π, which predicts the action chunk $\hat{\mathbf{A}}_t$ from the policy context and the predicted latent action $\hat{\mathbf{Z}}_t$, and jointly optimize policy parameters on embodiment-specific robot demonstrations with a combined loss:

$$\mathcal{L}_{\text{finetune}} = \mathcal{L}_{\text{BC}}\left(\hat{\mathbf{A}}_t,\, \mathbf{A}_t\right) + \lambda_z\, \text{SmoothL1}\left(\hat{\mathbf{Z}}_t,\, \text{sg}[\bar{\mathbf{Z}}_t]\right), \tag{10}$$

where λ_z balances behavior cloning against latent action regression. Because SpatialLAM assigns a fixed spatial identity to each token of $\bar{\mathbf{Z}}_t$, the auxiliary prediction loss provides the policy with a spatially indexed target rather than an unstructured transition representation.

Backbone. The policy follows the BAKU framework (Haldar et al., 2024). Each view is encoded by a shared ResNet-18 (He et al., 2016) with GroupNorm (Wu & He, 2018), trained from scratch and truncated after the second stage to 128 channels. Language conditioning is applied through FiLM (Perez et al., 2018) using frozen MiniLM-L6-v2 sentence embeddings (Wang et al., 2020). The resulting spatial feature map is pooled by a Spatial Softmax (Levine et al., 2016) with 256 keypoints and projected to a 512-dimensional token for each view. The language embedding and proprioception are each mapped to a 512-dimensional token by an MLP, and a learnable 512-dimensional action token is appended. A shared linear layer projects all tokens to width 256 and adds a learned positional embedding, after which a causal Transformer trunk with 8 layers and 4 heads fuses them into the policy context $\mathbf{c}_t$. Inputs are resized to 128×128, and the action chunk length is $T_a = 10$. Bridge pretraining uses a single camera view, whereas LIBERO fine-tuning uses both camera views.

Latent interface. The latent head h_z is a cross-attention decoder with 16 learned queries, width 256, 2 layers, and 4 heads. It attends to the policy context $\mathbf{c}_t$ and the ResNet feature maps $\mathbf{F}_t$ and predicts 16 latent tokens of width 4 in the same ordering as the corresponding LAM representation. For SpatialLAM, these tokens form the $4 \times 4 \times 4$ latent grid used by the regression targets in Equations 9 and 10.

The action decoder h_π first projects each latent representation to width 256 and adds the positional encoding associated with the corresponding latent interface. Ten learned horizon queries are then processed by one cross-attention block over the policy context followed by two cross-attention blocks over the 16 latent tokens. A two-layer MLP maps each horizon representation to a 7-dimensional action with a $\tanh$ output, producing a 10-step action chunk.

For the unstructured LAM, latent tokens are projected independently with a linear layer. For SpatialLAM, the 4×4 latent grid is projected with a 3×3 convolution with padding 1, allowing neighboring latent positions to interact before cross-attention.

Training. Pretraining on Bridge V2 optimizes Equation 9, and fine-tuning on LIBERO optimizes Equation 10. Table 7 gives both schedules, including the schedule of λ_z as the latent weight. The checkpoint transferred from pretraining is the one with the lowest held-out latent loss. Policy pretraining uses the full standard Bridge V2 training split, which contains 38,571 episodes and 919,192 transitions, and LIBERO supplies 50 demonstrations for each of the 10 tasks in each of the 4 suites.

The visual backbone and language projection use learning-rate multipliers of 0.1 and 0.5, respectively. The remaining policy modules use the base learning rate. Policy training applies color jitter with brightness, contrast, and saturation strength 0.1 and hue 0.02. No augmentation is applied at evaluation.

A.4 World Model

Formulation. A frozen pretrained encoder maps the current observation to a latent state feature $\mathbf{s}_t$. Transformer blocks apply self-attention over $\mathbf{s}_t$ and cross-attention to the latent action $\bar{\mathbf{Z}}_t$, predicting a residual update to the next state feature $\hat{\mathbf{s}}_{t+1}$:

$$\hat{\mathbf{s}}_{t+1} = \mathbf{s}_t + g_{\text{wm}}\big(\mathbf{s}_t,\, \bar{\mathbf{Z}}_t\big). \tag{11}$$

Table 7: Policy and world-model training settings.

	Policy		*World model*	
	Bridge pretrain	LIBERO fine-tune	One-step phase	Rollout phase
Steps	10 epochs	60k, 80k for Long	6,000	6,000
Global batch	2,048	128	128 clips	128 clips
Learning rate	10^{-4}, cosine	10^{-4}, cosine	$3\times10^{-4}\rightarrow3\times10^{-5}$, cosine	
Warmup	1,000	1,000	500	
Optimizer	AdamW	AdamW	AdamW	
Weight decay	0	10^{-4}	10^{-4}	
Gradient clip	1.0	1.0	1.0	
Latent weight	1.0 (sole term)	$1.0\rightarrow0.25$, cosine	–	
Pixel weight	–	–	0.05	0.10
Precision	bf16	bf16	bf16	
Hardware	$4\times$H100	$1\times$H100 per suite	$4\times$H100	

The model is optimized jointly with feature-space regression and pixel-space reconstruction objectives. At inference, we extract a sequence of latent actions from a reference video using the frozen LAM encoder and condition each autoregressive prediction step on the corresponding latent action, starting from the initial observation.

The world model operates in the same frozen standardized feature space as the LAM, with state $\mathbf{s}_t \in \mathbb{R}^{1024\times16\times16}$. The dynamics network g_{wm} has width 768, 8 Transformer blocks, 12 attention heads, an MLP ratio of 4, and no dropout. For SpatialLAM, the latent action is projected by a 3×3 convolution from 4 to 768 channels over the 4×4 latent grid. For the unstructured LAM, each latent token is projected by a linear layer. Both produce 16 conditioning tokens that are cross-attended by every Transformer block. The residual output projection of g_{wm} is initialized to zero.

Action-conditioning controls. Both controls use the same training objective, optimization schedule, and world-model transitions as the latent action variants. For *No Action*, we zero the 16×4 action tensor before the spatial projector, preserving the SpatialLAM conditioning pathway while removing transition-dependent action information. For *Recorded Robot Action*, we use the Bridge V2 control commands, each consisting of 3D translation, 3D rotation, and an absolute gripper state. Since one world-model transition spans ten raw control steps, the corresponding commands are concatenated into a 70-dimensional chunk. The seven command dimensions are standardized using statistics from the world-model training clips, covering 68,588 distinct commands from 745 Bridge V2 training episodes. A linear adapter projects the chunk to the same 16×4 conditioning interface, after which we use the unstructured LAM dynamics architecture.

Pixel decoder. Predicted feature maps are decoded to RGB by a frozen RAEv2 ViT-XL decoder (Singh et al., 2026) that was pretrained to invert the DINOv3 patch representation and is used here without modification. Each frame is decoded from its own predicted feature map.

Objective. For each transition, the feature-space objective combines a SmoothL1 loss on the predicted next-state features, a cosine-distance term weighted by 0.10, and a SmoothL1 loss on the predicted feature change weighted by 0.25. The pixel-space objective combines a Charbonnier reconstruction loss with an image-gradient loss weighted by 0.05 and an LPIPS loss weighted by 0.10.

Training proceeds in two phases. During the first phase, we perform one-step prediction by sampling one transition from each clip, with the pixel-space objective weighted by 0.05. During the second phase, the model is unrolled autoregressively for eight transitions from $\mathbf{s}_0$ without teacher forcing. The feature-space loss is averaged uniformly over all eight rollout steps, while the pixel-space objective is evaluated at one randomly sampled rollout step with weight 0.10. Table 7 summarizes the training schedule.

Data. For Bridge V2, clips contain 8 transitions sampled every 10 raw frames, requiring 81 raw frames per clip. This produces 9,733 training clips from 745 qualifying training episodes. Evaluation uses one deterministic 8-transition clip from each of 100 validation episodes.

For SSv2, we use 34,000 training clips sampled in a class-balanced manner over the 174 action classes. Each video is uniformly resampled over its full duration to 9 frames, corresponding to 8 transitions. Frames are center-cropped to a square and resized to 256×256. Evaluation uses 696 validation clips, with 4 clips per class. All world models read the unstructured LAM and SpatialLAM trained with seed 42. Both LAMs are trained only on the Open X-Embodiment mixture in Table 6 and never observe SSv2 videos, so the SSv2 latent actions are extracted zero-shot from human videos outside the LAM training distribution.

A.5 Evaluation Protocols

LIBERO. Each of the four suites contains 10 tasks evaluated with 50 rollouts per task, giving 2,000 rollouts in total. Episodes begin after 10 settling steps and are capped at 240, 300, 320, and 550 environment steps for Spatial, Object, Goal, and Long. The action execution horizon is 1, so the policy re-predicts at every control step under temporal aggregation with an exponential weighting coefficient of 0.01. Action postprocessing uses unit translation and rotation scales, a gripper threshold of 0.5, and one step of gripper stickiness.

LIBERO-Plus. Evaluation follows the official one-rollout-per-task protocol over 10,030 tasks, comprising 2,402 Spatial, 2,518 Object, 2,591 Goal, and 2,519 Long tasks. The same policy checkpoints, action execution protocol, and episode limits are used as in LIBERO. Each task carries exactly one of the seven perturbation categories.

Seeds. The unstructured LAM and SpatialLAM results in Tables 1 and 2 come from three runs with seeds $\{42, 43, 44\}$ for LAM pretraining, policy pretraining, and fine-tuning on each suite, and the policy of each run reads the LAM pretrained in the same run. The two methods use the same three seeds. Each run is evaluated on all 2,000 LIBERO rollouts and all 10,030 LIBERO-Plus tasks. For every column, including Avg. and Total, we compute the score of each run and report the mean and sample standard deviation over the three runs. Prior-method results are reported as published, without standard deviations.

Ablation subset. Ablation studies use a fixed stratified 10% subset of LIBERO-Plus (1,003 tasks). We sample 1,003 of the 10,030 evaluation tasks using stratified random sampling over the four LIBERO-Plus suites, perturbation categories, and difficulty levels, yielding 240 Spatial, 252 Object, 259 Goal, and 252 Long tasks. The subset is sampled once with seed 42 and reused for all ablation variants, ensuring that every comparison is evaluated on exactly the same tasks.

World-model metrics. PSNR and LPIPS are computed between predicted and ground-truth RGB frames at horizons $H \in \{1, 4, 8\}$. $H = 1$ denotes teacher-forced one-step prediction averaged over all eight transitions, and $H = 4$ and 8 denote autoregressive rollouts from $\mathbf{s}_0$. LPIPS (Zhang et al., 2018) is computed with the official `lpips` package (version 0.1), using the unmodified ImageNet-pretrained VGG-16 backbone and the released linear calibration weights. Each frame is center-cropped to a square, resized to 256×256 with bicubic interpolation, and linearly mapped to $[-1, 1]$, the input range expected by the package. The built-in input layer of the package converts this range to ImageNet-normalized values, so no separate normalization is applied. Feature NMSE is computed in the standardized DINOv3 feature space and normalized by a persistence predictor. For $H = 1$ the persistence predictor outputs the current ground-truth feature $\mathbf{s}_t$ as the prediction of $\mathbf{s}_{t+1}$. For $H \in \{4, 8\}$ it outputs the initial feature $\mathbf{s}_0$, giving

$$\mathrm{NMSE}_H = \frac{\|\hat{\mathbf{s}}_H - \mathbf{s}_H\|_2^2}{\|\mathbf{s}_0 - \mathbf{s}_H\|_2^2}, \tag{12}$$

where $\hat{\mathbf{s}}_H$ is rolled out autoregressively from $\mathbf{s}_0$. Evaluation clips are grouped by language instruction on Bridge V2 and by action class on SSv2. Within each group, the numerator and denominator are summed over clips before taking the ratio, and the reported NMSE is the unweighted mean of these per-group ratios. PSNR and LPIPS are likewise averaged within each group and then across groups.

FVD uses the RGB I3D network (Carreira & Zisserman, 2017) pretrained on Kinetics-400, with the PyTorch implementation and the checkpoint `i3d_pretrained_400.pt` released with TATS (Ge et al., 2022). Each video is represented by the pre-softmax logits averaged over time. Following the TATS evaluation, frames are resized from 256×256 to 224×224 with bilinear interpolation and scaled to $[-1, 1]$. Each generated video consists of the ground-truth conditioning frame followed by the 8 predicted frames of the $H = 8$ rollout, and each reference video consists of the same conditioning frame followed by the 8 corresponding ground-truth frames. On Bridge V2, the 9 frames are sampled every 10 raw frames, matching the world-model training stride, over 100 validation clips. On SSv2, the 9 frames are obtained by uniformly resampling each validation video over its full duration, over 696 validation clips with 4 clips per class. One Gaussian is fit to all generated videos and one to all reference videos, without per-group averaging.

A.6 Real-World Experiments

Hardware and sensing. The real-world platform consists of a single 7-DoF Flexiv Rizon 4 arm equipped with a Flexiv GN01 parallel gripper with an approximately 0.1 m stroke. The policy observes two Intel RealSense cameras that record 640×480 RGB images at 30 fps, namely a fixed third-person camera that faces the robot and covers the workspace, and a wrist-mounted camera. Each image is center-cropped to a square and resized to 128×128. Proprioception consists of the seven joint angles and the finger positions $[w/2, -w/2]$ derived from the measured gripper width w, without normalization.

Latent action targets. The LAM target $\bar{\mathbf{Z}}_t$ is computed from the fixed camera only, consistent with the single-camera target used on Bridge V2 and LIBERO. Both LAMs are the seed-42 runs of Appendix A.5, remain frozen, and are never trained on the real-world data. They encode the same 128×128 fixed-camera frames observed by the policy, upsampled to 256×256 with bilinear interpolation before the frozen DINOv3 encoder, so the LAM operates on the same 16×16 patch grid as in pretraining. Each policy is initialized from the Bridge V2 pretrained checkpoint of the seed-42 run, which reads the same LAM, and is then fine-tuned on the real-world demonstrations.

Action space. The policy predicts a 7-dimensional action, so the action decoder h_π retains the output dimension described in Appendix A.3. The first six dimensions represent the change in end-effector pose between consecutive recorded frames in the robot base frame. They comprise translation in meters and the relative rotation $R_{t+1}R_t^\top$ in axis-angle form, both computed from measured rather than commanded poses. The seventh dimension is derived from the gripper command $c \in [0, 1]$, which sets the gripper width to $0.1c$ m. Each pose dimension is linearly mapped from its 1st and 99th percentiles to $[-1, 1]$ and clipped to this range. Hang cup and Pick bread share percentiles computed from their pooled demonstrations, while Clean table uses task-specific percentiles. The gripper dimension is mapped to $1 - 2c$, following the LIBERO convention where -1 denotes open and $+1$ denotes closed, and remains continuous rather than binarized.

Inference. The policy predicts an action chunk of $T_a = 10$ steps, corresponding to 0.33 s at 30 Hz, and replans at every step. As in the LIBERO evaluation, the executed action aggregates all predictions that overlap the current step, up to 10 predictions, using exponential weighting with coefficient 0.01.

Demonstrations and fine-tuning. Demonstrations are collected by teleoperation using a 3Dconnexion SpaceMouse, which commands end-effector translation in the robot base frame. The robot controller runs at 100 Hz, while observations and poses are recorded at 30 Hz. Table 8 summarizes the demonstration data. A separate policy is trained for each task using all available demonstrations, with no held-out episodes. Each policy is fine-tuned with Equation 10 for 30,000 steps. All other optimization and augmentation settings, including the learning-rate and latent-weight schedules, follow the LIBERO configuration in Table 7. The two LAM variants use identical demonstrations, training schedules, augmentations, and random seeds.

Table 8: **Real-world tasks.** Demos denotes the number of teleoperated demonstrations used for fine-tuning. Length reports the mean demonstration duration, with the range shown in brackets.

Task	Instruction	Demos	Length (s)	Success criterion
Hang cup	Hang the cup on the rack.	30	17.2 [14.8, 20.6]	The cup hangs from the rack by its handle after release.
Pick bread	Pick up the bread and place it on the plate.	30	15.9 [13.5, 19.6]	The bread rests on the plate after release.
Clean table	Clean the table.	30	15.6 [12.7, 18.5]	The can rests inside the basket after release.

Evaluation protocol. Before evaluation, we sample 20 initial layouts per task uniformly from the region covered by the demonstrations. For Hang cup and Pick bread, the grasp and release positions each vary over an approximately 5×5 cm region. For Clean table, the can and basket are shifted together along the lateral axis of the robot base. Each layout is recorded as a reference image from the fixed camera, which the operator uses to restore the scene before every trial. Both policies are evaluated once on each layout. For a given layout, the two policies are executed consecutively in a randomized order concealed from the operator. All trials for a task are completed within a single session to minimize differences in lighting and hardware drift between policies. A trial terminates upon success, an unrecoverable failure such as dropping the object out of reach or triggering a safety stop, or reaching twice the longest demonstration duration for that task. This yields time limits of 41, 39, and 37 s for Hang cup, Pick bread, and Clean table, respectively. The operator resets the scene between trials and does not intervene during execution. Success is determined at the end of each trial according to the criteria in Table 8.

Displaced basket. In the Clean table demonstrations, the can and basket are shifted together along the lateral axis of the robot base, with the basket kept to the right of the can. Across demonstrations, the end-effector grasp position spans 9.6 cm laterally and 1.9 cm forward, while the release position spans 12.5 cm laterally and 2.5 cm forward. The layout variation is therefore concentrated along the lateral axis. For the displaced evaluation, the can is fixed at the center of its training range, while the basket is translated by 5 cm from its training position relative to the can. We evaluate ten translation directions evenly spaced at 36° in the table plane, starting from the lateral axis of the robot base, with one trial per direction. Each displacement therefore changes the can-to-basket placement, which the demonstrations preserve when shifting the layout. Apart from the two lateral directions, every direction also has a forward or backward component of at least $5 \sin 36^\circ \approx 2.9$ cm, which exceeds the entire 2.5 cm forward range observed at release during training. Both policies are evaluated on the same ten basket positions using the protocol above.

A.7 Model Size and Training Resources

Table 9: **Measured parameter counts.**

Module	Unstructured LAM	SpatialLAM
LAM	68,292,105	68,292,105
Policy latent head	2,144,004	2,144,004
Policy action decoder	3,432,455	3,440,647
World-model dynamics	77,980,672	78,005,248

The two LAM variants have identical trainable parameter counts. The small parameter increase in the downstream models arises solely from replacing the per-token linear latent projection with a 3×3 convolutional projection for SpatialLAM. This adds 8,192 parameters to the policy action decoder (0.24%) and 24,576 parameters to the world-model dynamics (0.03%). All remaining components are unchanged between the corresponding downstream models. Table 9 summarizes the parameter counts.

LAM training uses four H100-80GB GPUs. Policy pretraining uses four H100 GPUs, while each LIBERO suite is fine-tuned on a single H100. World-model training uses four H100 GPUs. The corresponding batch sizes and optimization schedules are reported in Tables 5 and 7.

B Ablation Protocols

B.1 Latent Action Visualization

Figure 1 uses the LIBERO-Long task *turn on the stove and put the moka pot on it* in the KITCHEN_SCENE3 layout. To vary the spatial location of an action while preserving the scene configuration, we rigidly translate the workspace in the horizontal plane with the agent-view camera fixed, placing the end effector at each cell of a 4×4 grid over the 128×128 image. We evaluate 10 translation actions at all 16 locations, yielding 160 action-location pairs. For each transition, we extract the 16×4 posterior-mean latent action and standardize each token channel using statistics computed from Bridge V2 latents. Token activation is computed as the difference between the latent for the action transition $(\mathbf{o}_t, \mathbf{o}_{t+1})$ and that for the corresponding static pair $(\mathbf{o}_t, \mathbf{o}_t)$. Finally, we center the activation of each token by its mean over all 160 transitions and retain the positive component for visualization.

B.2 Component and Control Ablation

Shared evaluation protocol. All policy variants in Table 4 are evaluated on LIBERO-Long and on the same fixed 1,003-instance subset of LIBERO-Plus described in Appendix A.5. The subset is sampled once with seed 42 using stratification over suite, perturbation category, and difficulty, and is held fixed across all variants to ensure a controlled comparison. World-model performance is measured on Bridge V2 and SSv2 under the protocol of Appendix A.5, identical for all variants.

Variants. The component ladder in Table 4 progressively augments the unstructured LAM with the spatial grid, locality bias, and convolutional projector, allowing the contribution of each component to be isolated under otherwise matched settings. The No latent control removes the latent head h_z and the latent action objective, so the policy regresses the action chunk directly from the policy context. The Unstructured + conv. projector control adds only the convolutional projector to the unstructured baseline, testing whether the gain can be explained by the projector itself rather than by spatial organization. The Shuffled grid + locality bias variant retains the locality bias but removes a persistent token-to-region mapping. We draw 16 query-to-anchor permutations once with a fixed seed. For each input pair, we sum $\|\mathbf{V}_{t+1,mn} - \mathbf{V}_{t,mn}\|_2^2$ over the 4×4 patches of each latent-grid cell and use the permutation indexed by the cell with the largest change. In the encoder, the positional encoding and locality bias of each query follow its assigned anchor. The decoder does not receive the assignment and reads the latent tokens in query order, so the location of the largest change reaches the FDM only through the latent. Because the assignment is a deterministic function of the input pair, each transition has a single latent target, as in the unstructured LAM. Resampling the assignment randomly for each sample would instead map the same transition to different latent targets and confound the absence of persistent correspondence with target noise. In Table 4, downstream models read this variant with the per-token linear projector of the + locality bias variant. OTF-LAM (Nam et al., 2026) and PoLAR (Jeong et al., 2026) are evaluated using their official implementation and trained with the same data, optimization, and latent capacity as our variants.

FLOPs accounting. We report the computational cost of latent inference per two-frame sample, corresponding to the computation required to extract latent actions for downstream use. The accounting includes the input projection, all eight encoder blocks, and the posterior heads, while excluding the frozen DINOv3 visual encoder (Siméoni et al., 2025) and the reconstruction decoder, which is used only during LAM training. We count each multiply-accumulate in linear and convolutional layers as two FLOPs and each elementwise addition as one FLOP. In SpatialLAM, the locality bias introduces one additional elementwise addition for each latent-patch attention logit. With 16 latent tokens and 256 visual patches in both attention directions, this amounts to $8{,}192$ additions per attention head and encoder block, or 1.05M FLOPs across eight heads, two frames, and eight blocks. The 3×3 spatial projector contributes a further 0.26M FLOPs relative to the per-token

linear projection used by the unstructured baseline. Table 10 summarizes the resulting parameter and FLOP overhead for each variant.

Table 10: **Parameter and FLOP overhead of controlled variants.** Values are reported relative to the unstructured LAM baseline, counting the LAM together with the downstream latent projector. The baseline contains 68.293M parameters and requires 39.490G FLOPs per two-frame sample.

Method	ΔParams	ΔFLOPs
+ spatial grid	0	0
+ locality bias	0	1.05M
+ conv. projector (SpatialLAM)	8.2K	1.31M
Unstructured + conv. projector	8.2K	0.26M
Shuffled grid + locality bias	0	1.05M
OTF-LAM	0.54M	18.68G
PoLAR	−1.97K	13.26G

B.3 Cross-position Spatial Generalization

Environment construction. The benchmark starts from all 10 LIBERO-Long tasks and their 500 original demonstrations. We construct spatial variants in the simulator by applying the same planar offset to the robot base, task-relevant fixtures, and movable objects, preserving their relative geometry and replaying the original action sequence with the external agent-view camera fixed. The 4×4 workspace lattice spans ± 9 cm along each tabletop axis, with 6 cm between adjacent positions. We retain a replay only when the original LIBERO success predicate is satisfied and the interaction remains in view. Among source demonstrations valid at all 16 positions, we uniformly retain 40 per task, yielding 400 source demonstrations.

Position split and training budgets. A checkerboard partition divides the lattice into eight candidate training positions and eight fixed test positions. Policies use nested subsets of $k \in \{1, 2, 4, 8\}$ candidate training positions, and every setting is evaluated on the same eight held-out positions. The $k = 16$ reference includes all positions during training, so its evaluation is in distribution. Each LAM is pretrained once and frozen across these settings.

Under the matched budget, the total is fixed at 400 demonstrations, distributed evenly over the selected positions: 400, 200, 100, 50, and 25 per position for $k = 1, 2, 4, 8, 16$, respectively. Under the natural budget, all 400 source demonstrations are instantiated at every selected position, giving $400k$ training demonstrations. The matched budget isolates spatial coverage at a fixed demonstration count. The natural budget varies both coverage and data volume.

Correspondence controls and evaluation. Both controls keep the locality bias and the 3×3 convolutional projector of SpatialLAM and differ from it only in the query-to-anchor assignment. Shuffled grid uses the transition-dependent assignment described in Appendix B.2. Unlike the shuffled-grid variant in Table 4, which downstream models read with the per-token linear projector, it is read here with the 3×3 projector. Fixed permutation uses a random assignment held constant across samples, retaining a persistent association while scrambling its spatial arrangement. Both controls are reported under the matched budget in Figure 4. Each condition is evaluated with 10 rollouts per task at each of the eight test positions, for 800 rollouts.

B.4 Persistent Spatial Correspondence of Latent Tokens

Objective and experimental setup. We ask whether individual latent tokens exhibit localized spatial effects and, more importantly, whether these spatial roles remain consistent across transitions. We compare the unstructured LAM and SpatialLAM on 64 held-out Bridge V2 transitions. For each transition, we intervene on one latent token at a time by replacing it with the same-index token from another held-out transition, while keeping the current-frame visual representation and the remaining 15 latent tokens fixed. The modified latent is then decoded by the frozen FDM, and its patchwise reconstruction change is measured relative to the unperturbed output. Repeating this procedure over all 16 latent tokens yields 1,024 influence maps per model.

Residual influence maps. To isolate token-specific spatial effects from reconstruction changes shared across token interventions, we subtract the mean influence map over the 16 interventions within each transition. We retain the positive residual and normalize it to unit mass. The resulting influence is then aggregated over 16 non-overlapping visual regions, each covering 4×4 patches and corresponding to one cell of the 4×4 latent grid. These normalized maps characterize where each latent token exerts its relative spatial influence, rather than the absolute magnitude of the induced reconstruction change.

Metrics. We distinguish spatial locality within a transition from correspondence across transitions. Local concentration measures the residual mass contained in the most affected visual region, selected independently for each intervention, and therefore quantifies whether the effect of a token is spatially localized without requiring a fixed spatial assignment. Cross-sample correspondence measures whether this spatial role persists across transitions: for each token, we estimate its preferred region from the other 63 transitions and evaluate the residual mass assigned to that region in the held-out transition, repeating this leave-one-out procedure over all 64 transitions. Anchor alignment measures the mass contained in the region prescribed by the corresponding SpatialLAM anchor. It is undefined for the unstructured LAM, which has no predetermined spatial assignment. Table 11 reports these metrics together with permutation-based null references.

Table 11: **Spatial correspondence of latent-token interventions.** Scores aggregate 1,024 interventions over 64 held-out Bridge V2 transitions per model, using normalized positive residual influence maps. Local concentration measures the mass in the most affected region for each intervention. Cross-sample correspondence measures the mass in a token-specific region estimated from the other 63 transitions. Anchor alignment measures the mass in the region prescribed by the SpatialLAM anchor and is undefined for unstructured LAM. Higher scores indicate stronger concentration or correspondence. Null values are permutation references.

Metric	Unstructured LAM		SpatialLAM	
	Score	Null	Score	Null
Local concentration	0.408	0.181	**0.505**	0.189
Cross-sample correspondence	0.184	0.092	**0.363**	0.111
Anchor alignment	–	–	**0.359**	0.063

Results and interpretation. Table 11 shows that the unstructured LAM already exhibits substantial spatial locality: its local concentration reaches 0.408, compared with a permutation null of 0.181. However, this locality does not translate into a stable token-to-region correspondence, consistent with the sample-dependent spatial effects visualized in Figure 5. SpatialLAM increases local concentration to 0.505 and, more importantly, raises cross-sample correspondence from 0.184 to 0.363, compared with permutation nulls of 0.092 and 0.111 for the two models, respectively. Because the preferred region of each token is estimated exclusively from other transitions, this increase reflects greater consistency of its spatial role across samples rather than within-sample localization alone. SpatialLAM further achieves an anchor alignment of 0.359 against a null of 0.063, indicating that these stable spatial roles agree with the prescribed latent grid. Together, these results show that unstructured LAMs can develop localized spatial effects without explicit structure, whereas SpatialLAM organizes these effects into persistent token-to-region correspondence across transitions.

C Additional Ablation Studies

C.1 Latent Capacity and Auxiliary Supervision

Grid resolution. Table 12 varies the number of latent tokens L_z while fixing the per-token width at $d_z = 4$, and compares each spatial grid with an unstructured LAM of matched latent capacity. SpatialLAM consistently outperforms its matched unstructured baseline across all three capacities. With a 2×2 grid, SpatialLAM improves success by 6.0 pp on LIBERO-Long and 16.8 pp on the LIBERO-Plus subset, although both variants remain below their 16-token counterparts. At $L_z = 64$, SpatialLAM remains strong with an 8×8 grid, while the corresponding unstructured baseline

degrades relative to the 16-token setting. We therefore use the 4×4 grid as the default, as it achieves the best LIBERO-Long performance and remains within 0.5 pp of the 8×8 grid on the LIBERO-Plus subset.

Table 12: **Grid-resolution ablation.** Success rate (%) on LIBERO-Long and the LIBERO-Plus subset. Each spatial configuration is compared with an unstructured LAM of matched latent capacity, with per-token width fixed at $d_z = 4$.

Config	L_z	Total dim.	Long	Plus
Unstructured LAM	4	16	63.4	42.4
2×2	4	16	69.4	59.2
Unstructured LAM	16	64	76.8	63.2
4×4	16	64	83.0	69.6
Unstructured LAM	64	256	71.6	57.1
8×8	64	256	82.6	70.1

Action alignment. Table 13 evaluates the effect of auxiliary action alignment on both the unstructured LAM and SpatialLAM. Action alignment improves both models, increasing success by $2.6/3.7$ pp for the unstructured LAM and $3.4/2.5$ pp for SpatialLAM on LIBERO-Long/Plus, respectively. Importantly, SpatialLAM consistently outperforms the unstructured baseline both with and without action alignment, showing that its advantage does not depend on this auxiliary supervision.

Table 13: **Action-alignment ablation.** Success rate (%) on LIBERO-Long and the LIBERO-Plus subset, with and without auxiliary action alignment.

Configuration	Long	Plus
Unstructured LAM	74.2	59.5
+ action alignment	76.8	63.2
SpatialLAM	79.6	67.1
+ action alignment	**83.0**	**69.6**

Table 14 further examines sensitivity to the action-alignment weight λ_a in Equation 5. The default value of 10^{-2} achieves the best performance on both benchmarks. Increasing λ_a to 10^{-1} reduces success by 1.6 pp on LIBERO-Long and 2.1 pp on the LIBERO-Plus subset, while decreasing it to 10^{-3} results in smaller drops of 1.0 and 0.2 pp, respectively. SpatialLAM remains above the unstructured baseline across the full range of tested weights, indicating that its advantage is robust to moderate variation in the strength of action alignment.

Table 14: **Sensitivity to the action-alignment weight.** Success rate (%) on LIBERO-Long and the LIBERO-Plus subset.

Configuration	Long	Plus
Unstructured LAM	76.8	63.2
SpatialLAM, $\lambda_a = 10^{-1}$	81.4	67.5
SpatialLAM, $\lambda_a = 10^{-2}$	**83.0**	**69.6**
SpatialLAM, $\lambda_a = 10^{-3}$	82.0	69.4

Locality scale. Table 15 examines the sensitivity of SpatialLAM to the locality scale ρ in Equation 7, measured in units of latent-grid spacing. For the default 4×4 latent grid over a 16×16 visual patch grid, $\rho = 1$ corresponds to one latent-grid spacing, or four visual patches. This setting achieves the best performance on both LIBERO-Long and the LIBERO-Plus subset. Performance

degrades moderately when the locality prior is either substantially narrower or broader, but remains above the unstructured baseline throughout the tested range. Overall, SpatialLAM is robust to moderate variation in ρ, with the best results obtained when the locality scale is matched to the spacing of the latent grid.

Table 15: **Locality-scale ablation.** Success rate (%) on LIBERO-Long and the LIBERO-Plus subset. The locality scale ρ is measured in latent-grid spacings. For the default 4×4 grid, $\rho = 1$ corresponds to four visual patches.

ρ	Long	Plus
0.25	80.4	66.9
0.5	82.0	68.8
1	83.0	69.6
2	79.6	67.8

C.2 Cross-Position Action Retrieval

Transition collection. We construct the retrieval probe from 12 restored states of a LIBERO-Long task. For each state, the scene is translated such that the end effector projects to each cell of a 4×4 grid in the agent-view image plane. At each location, we execute one of eight unit-normalized translation directions in the y-z plane, $\{\pm y, \pm z, \pm y \pm z\}$, for eight open-loop steps while keeping the gripper open. We discard transitions for which the end effector leaves the field of view or its displacement is below 1 cm, resulting in 1,347 valid transitions with approximately balanced action classes.

Queries and candidate galleries. Each retained transition is used once as a query. To avoid trivial matches from temporally adjacent or near-duplicate observations, we exclude candidates originating from the same source demonstration and restrict retrieval to transitions from different restored states. For same-position retrieval, the candidate gallery contains only transitions at the query location. For cross-position retrieval, candidates at the query location are excluded. The resulting evaluation is a nearest-neighbor probe of the learned representation and introduces neither a learned retrieval head nor a supervised train-test split.

Table 16: **Cross-position action retrieval.** Top-1 nearest-neighbor retrieval accuracy across 1,347 LIBERO-Long transitions spanning eight action classes. Candidates are drawn from different restored states. Fixed-order concatenates all latent tokens in a fixed order, and location-aligned uses the 3×3 latent window centered on the end-effector cell. Chance accuracy is 0.125.

Model	Readout	Same location	Other locations
Unstructured LAM	fixed-order	0.72	0.35
Unstructured LAM	location-aligned	0.62	0.18
SpatialLAM	fixed-order	**0.81**	0.21
SpatialLAM	location-aligned	**0.81**	**0.62**

Representation and retrieval. We use the deterministic posterior mean as the latent representation and standardize each token channel using statistics computed over the retained probe set. Candidates are ranked by cosine similarity, and retrieval is considered correct when the top-ranked candidate has the same action class as the query. We report Top-1 accuracy micro-averaged over all queries. With eight approximately balanced action classes, uniform chance performance is 12.5%. We apply two readouts to both models. The fixed-order readout concatenates all 16 latent tokens in a fixed order, and therefore rewards a representation that stays unchanged when the action moves. The location-aligned readout arranges the tokens on the 4×4 grid and extracts the 3×3 window centered at the cell of the end effector, with zero padding outside the grid, and therefore rewards a representation that moves with the action.

Table 17: **Shift equivariance of latent actions.** Same-action cosine similarity and action-specific margin between posterior-mean latents at two image locations, computed after shifting the latent grid by the image-cell offset (aligned) or in the fixed token order (unaligned).

Model	Same-action cosine		Margin	
	Aligned	Unaligned	Aligned	Unaligned
Unstructured LAM	0.042	0.081	0.025	0.077
SpatialLAM	**0.580**	0.024	**0.214**	0.041

Within the same location, both models retrieve the action class reliably (Table 16). Across locations, SpatialLAM with the location-aligned readout reaches 0.62, compared with 0.35 for the best readout of the unstructured LAM. The two models respond to the readouts in opposite ways. The unstructured LAM drops to 0.18 under the location-aligned readout, since its token order carries no spatial meaning. SpatialLAM drops to 0.21 under the fixed-order readout, since translating an action shifts its encoding across the latent grid (Appendix C.3). SpatialLAM therefore encodes an action consistently across locations, at the grid position where the interaction occurs. Downstream models read the latent through a 3×3 convolutional projector that applies the same local operation at every grid position, and the cross-position policy results in Figure 4 show that the policy benefits from this organization.

C.3 Shift Equivariance of Latent Actions

We test whether translating an interaction translates its latent grid, using the probe of Appendix C.2. For two transitions of the same restored state and action class at image cells $\mathbf{c}_i$ and $\mathbf{c}_j$, we shift the 4×4 latent grid of the first by $\boldsymbol{\delta} = \mathbf{c}_j - \mathbf{c}_i$ and compute its cosine similarity with the second on the tokens that remain inside the grid, without padding. We also compare the two latents in their fixed token order. Because rigid workspace translation also shifts the static scene, we subtract the similarity between transitions with the same state and cells but different action classes, and report this margin as the action-specific component. Both comparisons are applied to both models over 9,481 same-action pairs.

Shifting the SpatialLAM grid by the image offset recovers the latent at the other location with cosine 0.580, compared with 0.024 in the fixed token order, so its encoding moves across the grid together with the interaction (Table 17). For the unstructured LAM, the fixed order gives the higher similarity, and shifting lowers it from 0.081 to 0.042, since its token order carries no spatial meaning. Under the better comparison for each model, the action-specific margin is 0.214 for SpatialLAM and 0.077 for the unstructured LAM.

C.4 Downstream Reliance on Latent Actions

We test whether the downstream models functionally rely on the learned latent actions rather than bypassing them through observation-only shortcuts. At inference, we intervene only on the latent input and keep the observations, model parameters, and all other inputs unchanged. For the policy, we replace the predicted latent with the training-set mean, Gaussian noise, a latent from another task in the same suite, or a fixed permutation of the 16 latent token positions. Each policy is evaluated under five conditions: these four interventions and the unperturbed policy. We re-run the unperturbed policy at the same budget, so every intervention is paired with it episode by episode from the same initial state. Each condition uses 100 rollouts per suite (10 initial states for each of the 10 tasks) across all four LIBERO suites. This gives 2,000 closed-loop rollouts per policy and 4,000 across the two policies. For the world model, we replace the matched conditioning latent with a no-motion latent or a latent from a same-class, cross-class, or opposite-motion clip, and evaluate eight-step rollouts on the same 696 SSv2 clips.

As shown in Table 18, both downstream models degrade substantially when their latent inputs are intervened upon. For the policy, every intervention reduces success by at least 25.5 pp. Token permutation is particularly informative because it preserves all latent values while changing only their positional assignment. It reduces success by 38.0 pp for SpatialLAM, compared with 25.5 pp

Table 18: **Downstream dependency on latent actions.** (a) Closed-loop LIBERO success rate (%). All five conditions share the same 400 initial states per policy (100 per suite), so each intervention is paired with *Own latent* episode by episode. *Own latent* re-runs the unperturbed seed-42 policy on the same 400 initial states, so it differs slightly from the three-seed averages in Table 1. *Mean latent* replaces the latent action with the dataset mean. *Gaussian noise* samples a new noise latent action at every control step. *Cross-task latent* uses a latent action from another task in the same suite. *Grid permutation* applies a fixed token permutation, preserving their contents while changing their spatial assignment. (b) LPIPS after eight-step rollouts on 696 SSv2 validation clips. *No-motion* uses the latent action encoded from an unchanged pair $(\mathbf{o}_t, \mathbf{o}_t)$. *Same-class* and *cross-class* use latent actions encoded from another clip of the same or a different action class. *Opposite-motion* uses a latent action encoded from a clip with the opposite global motion direction. Δ denotes the change relative to the unperturbed latent action for that model. Lower success rate and higher LPIPS indicate stronger degradation.

(a) Policy, closed-loop LIBERO

Latent	Unstructured		SpatialLAM	
	SR	Δ	SR	Δ
Own latent	92.5	–	94.8	–
Mean latent	31.8	-60.7	36.0	-58.8
Gaussian noise	27.3	-65.2	41.8	-53.0
Cross-task latent	26.3	-66.2	39.3	-55.5
Grid permutation	67.0	-25.5	56.8	-38.0

(b) World model, SSv2 eight-step rollout

Latent	Unstructured		SpatialLAM	
	LPIPS	Δ	LPIPS	Δ
Own latent	0.388	–	0.371	–
No-motion	0.435	$+0.047$	0.434	$+0.063$
Same-class	0.497	$+0.109$	0.507	$+0.136$
Cross-class	0.508	$+0.120$	0.519	$+0.148$
Opposite-motion	0.559	$+0.171$	0.567	$+0.196$

for the unstructured LAM, indicating greater sensitivity to positional disruption. This connects the downstream dependence to the image-space organization of SpatialLAM.

The world model similarly depends on its latent conditioning: all four replacements increase LPIPS, with an average increase of 0.136 for SpatialLAM and 0.112 for the unstructured LAM. The degradation also grows with stronger motion mismatch: same-class replacements increase LPIPS by $0.109/0.136$, cross-class replacements by $0.120/0.148$, and opposite-motion replacements by $0.171/0.196$ for the unstructured LAM and SpatialLAM, respectively. These interventions show that the latent actions are functionally used by both downstream models, while SpatialLAM exhibits greater sensitivity to disrupting the positional organization of its latent tokens.

C.5 World-Model Rollout Visualization

Figure 6 presents eight-step autoregressive rollouts of world models conditioned on latent actions extracted by the unstructured LAM and SpatialLAM. Both models start from the same conditioning frame and are driven by latent actions inferred from the same ground-truth transitions. In the SSv2 episode, the rollout conditioned on the unstructured LAM quickly loses the appearance and spatial trajectory. In contrast, SpatialLAM preserves its texture while maintaining its position close to the ground-truth trajectory. In the Bridge V2 episode, SpatialLAM more faithfully preserves the arm and gripper structure over later prediction steps, whereas the unstructured LAM exhibits increasing blur and spatial drift around the moving manipulator. These examples illustrate the benefit of spatially structured latent actions for world modeling: by maintaining persistent correspondence between latent tokens and local visual regions, SpatialLAM provides more spatially coherent transition signals, helping the world model preserve localized motion and object structure over long autoregressive rollouts.

C.6 Failure Cases

Global camera motion is challenging for image-plane latent actions because apparent motion mixes camera egomotion with object motion (Ye et al., 2025; Nikulin et al., 2025; Lee et al., 2026). As shown in Figure 7, such motion can shift the visual support of latent grids and accumulate as spatial

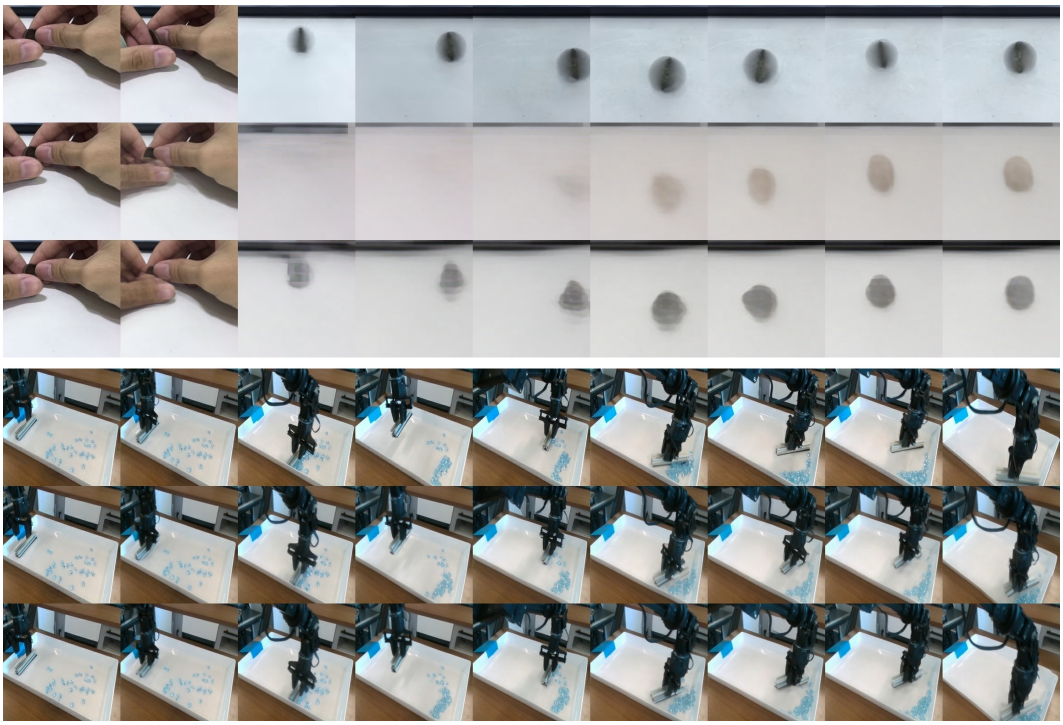


Figure 6: **Qualitative world-model rollouts on SSv2 (Goyal et al., 2017) and Bridge V2 (Walke et al., 2023).** Columns show the conditioning frame followed by autoregressive rollout steps $H = 1$ to 8. The top block shows the SSv2 episode, and the bottom block shows the Bridge V2 episode. Rows 1 and 4 are the ground-truth videos, rows 2 and 5 are rollouts conditioned on the unstructured LAM, and rows 3 and 6 are rollouts conditioned on SpatialLAM.

drift over long rollouts. In the displayed examples, SpatialLAM nevertheless preserves local motion structure more faithfully than the unstructured LAM under the same perturbation.

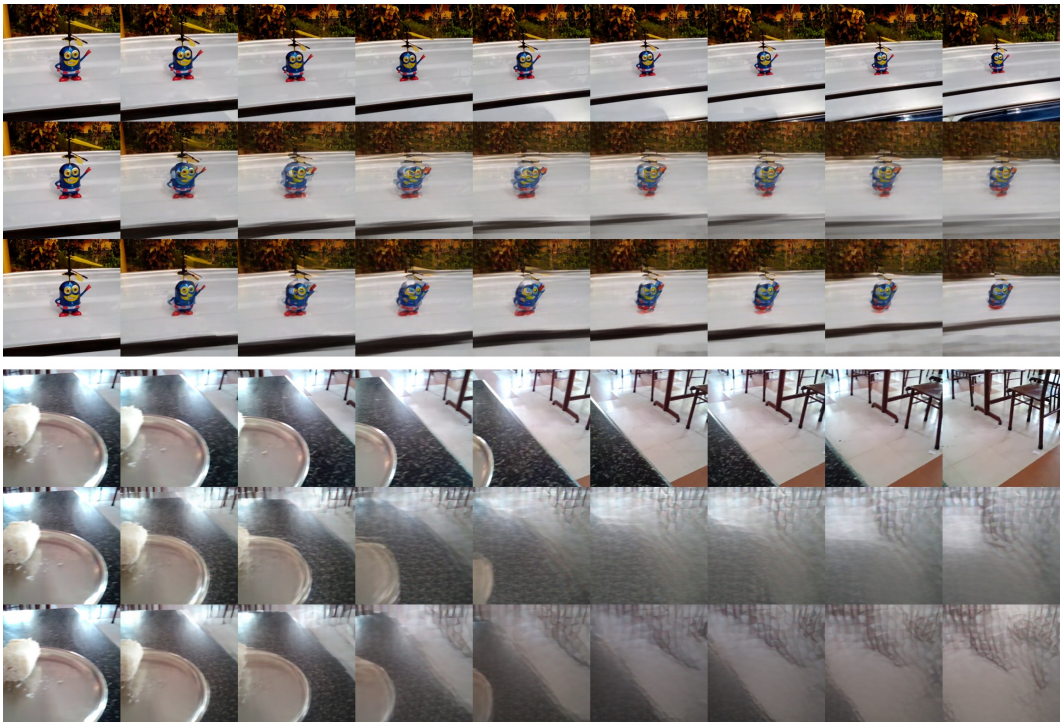


Figure 7: **Failure cases under camera motion.** Columns show rollout steps 0 to 8. Rows 1 and 4 are the ground truth, rows 2 and 5 are rollouts conditioned on the unstructured LAM, and rows 3 and 6 are rollouts conditioned on SpatialLAM. Global camera motion leads to accumulated spatial drift in long-horizon rollout, although SpatialLAM remains more spatially coherent than the unstructured LAM.